2008 International Symposium on Computer Science and Computational Technology

An Adaptive Development Framework for Web-based Enterprise Information System

Liu Xudong, Xu Xiaofei, Zhan Dechen, Qiao Limin
Department of Computer Science and Technology
Harbin institute of Technology
Harbin, China
{cameran, xiaofei, Dechen, qlm}@hit.edu.cn

Abstract—**The rapid evolutions of software environment and business requirements place a high demand on the adaptiveness of enterprise information systems (EIS). Over the last few years, more and more EIS adopted the distributed multi-tiered web-based application architecture. Crosscutting concerns and clone codes make the web-based EIS difficult to evolve and maintain. The traditional Model-View-Controller (MVC) model imposes design-level restrictions that give a clean separation between the presentation, functional and control concerns. However it does not modularize the structure crosscutting concerns. In this paper, we propose a novel development model named Meta Data Object (MDO) that modularizes the structure concerns and reduces some specific clone codes. Panther is a new domain-driven web development framework that implements the MDO model. Panther has been used to develop many web-based EIS. Development with Panther benefits from a significant improvement in code reuse, adaptability, and maintainability.**

Keywords-enterprise information system; web-based application; development framework; crosscuting concerns; maintainability; reusability

I. INTRODUCTION

The rapid evolutions of software environment and business requirements place a high demand on the adaptability of enterprise information system (EIS). Over the last few years, more and more EIS adopted the distributed multi-tiered web-based application architecture. So how to design and develop the adaptive web-based EIS also has become a hot research topic in recent years.

Currently, there are two main challenges in development of the adaptive web-based EIS. One is separation of concerns. The distributed multi-tiered architecture web application are normally comprised of a presentation tier to render graphical user interfaces (GUI), a functionality tier to process business logic, and a data tier to store configurations and business data. Consequently, web development concern including presentation, functionality, control, and structure cross-cut, leading to tangled and scattered code that is hard to develop, maintain, and reuse [1]. The traditional Model-View-Controller (MVC) model [2] imposes design-level restrictions that give a clean separation between the presentation, functional and control concerns. However, it does not address a more severe code scattering across the three tiers because of the structure crosscutting [1].

Another challenge is the clone codes. The clone code is a code portion in source files that is identical or similar to another [3]. In web-based EIS, there are amount of clone codes because of the similar user interfaces and similar business logics. When system requirements change, it is hard to modify the clone codes consistently. So the clone codes make the web-based EIS hard to evolve.

The contribution of this paper is introducing a novel approach for developing the web-based EIS. The core of the approach is the Meta Data Object (MDO) model that can separate the structure concerns and reduce some specific clone codes. And a new domain-driven web development framework has been implemented in the MDO model.

The remainder of the paper is organized as follows. In section 2, we give some background to crosscutting concerns and clone codes in web-based EIS. Section 3 presents a solution to these problems, and an overview of the implementation is also presented. Section 4 introduces the related works. Section 5 discusses the benefits and limitations of the approach, and Section 6 concludes the paper.

II. BACKGROUND

A. Web Application

The web-based applications are characterized by an unprecedented mix of features that makes them radically different from previous applications of information technology:

(1) Browser/server architecture. The web-based applications are browser/server software, which are normally comprised of a presentation tier to render graphical user interfaces, a logic tier to process business logic, and a data tier to store configuration and business data. In particular, the HTML standard is a cornerstone of the web, which defines forms to capture and submit user input.

(2) Hybrid programming. A web application is typically written in at least two programming languages. The presentation is described in one or more client-side languages (e.g., HTML, XML, JavaScript, Cascading style

sheets). The functionality is specified using a server-side language (e.g., Perl, Python, ASP, JSP, Java, C, and Smalltalk). So building web applications is a complex and time-consuming process.

(3) Dynamic web pages. The web user interfaces are often generated on the fly, which makes application code harder to understand and makes troubleshooting more difficult because of the extra level of abstraction that one must consider.

These features make the web-based EIS more difficult to maintain.

B. *Structure Crosscutting*

Separation of concerns is one important object of software engineering. There are four typical application concerns in the web-based applications: functionality, presentation, control and structure.

The functionality concern is the business logic that specifies the set of server-side operations to be performed upon receipt of a client request.

The presentation concern is the “look and feel” of the page. It is the user-interface that a web application provides to its clients.

The control concern specifies high-level control flow decisions. Based on the request and server state, the control logic defines what action to take next. It manages both functionality and presentation.

The structure concern refers to the entity of the business objects. It is the data used by the presentation and the functionality. The presentation maps the data to the user interfaces, and the functionality performs the operation on the data. And the concern includes the data transfer among the three tiers.

The MVC model imposes design-level restrictions that give a clean separation between the presentation, functional and control concerns by organizing the web application code in three modules, namely model, view, and controller. However, it does not modularize the structure crosscutting concerns perfectly. There are still many structure dependent code fragments in the MVC model. Especially, the view components generate user interface based on the data in the model, so when the data structure of the business model changes, the view and model components will be modified together.

C. *Clone codes*

In web-based EIS, there are amount of clone codes because of the similar user interfaces and similar business logics.

In EIS, the business operations of the business objects can be classified to two categories: generic operations and specific operations. For example, create, retrieve, update and delete (CRUD) operations are generic operations. Almost all the business objects have CRUD operations. The specific operations only belong to one specific business object. Obviously, there are amount of clone codes because of the generic operations in EIS.

The similar user interfaces root from the user interface design patterns [4]. For example, the Master/Detail Pattern is a typical pattern for business applications. It is also known as Master/Slave Pattern or Director/Details Pattern. The user interfaces with this pattern have two areas at least, one area is to display the main information unit (the master), the others are to display the detail information units (the slave), and the master determines the slave. In EIS, there are many user interface design patterns. Due to space limitations, they cannot be discussed in this paper.

III. AN ADAPTIVE DEVELOPMENT FRAMEWORK

A. *The Meta Date Object Model*

Separating the structure concerns and reducing the clone codes are crucial for building web-based EIS more efficiently. In this section, we present a novel development model, namely MDO, to resolve these problems.

Traditionally, the data transfer object (DTO) model [5] is used to modularize the structure concerns, and it has been adopted in many popular MVC frameworks, such as Struts [6], Spring [7], etc. DTO is a pure data object only has some getter and setter operations. DTO can separate the structure concern from the model component in MVC model by automating the object-relational (O-R) mapping, but it dose not separate the structure concern from view component.

The MDO model can easily separate the structure concern from view component by providing the user interface and relational (UI-R) mapping which is transparent for the developer. As illustrated in Figure 1, MDO model is easier to implement the O-R mapping than DTO model because the meta-data can be used directly.

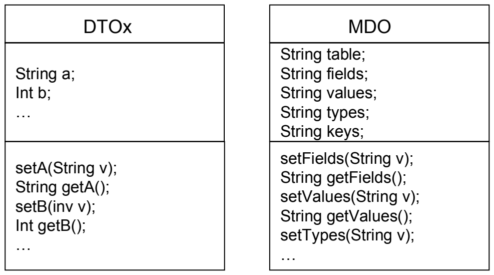

Figure 1. DTO and MDO

In the DTO model, every table has a class to implement the O-R mapping; the MDO model is based on instances rather than classes, there is only one class to implement both the O-R mapping and the UI-R mapping, so the extra advantage of MDO model is the low volume of the source code.

B. *The Panther Framework*

The Panther framework is a concrete implementation of the MDO model, and it is a domain-specific framework which supports the patterned user interfaces and distinguishes the generic and specific business operations.

The architecture of Panther framework is shown in Figure 2. The framework is comprised of six components, namely, front controller (FC), generic business logic component (GBLC), specific business logic component

(SBLC), user Interface component (UIC), data block component (DBC) and database wrapper (DBW).

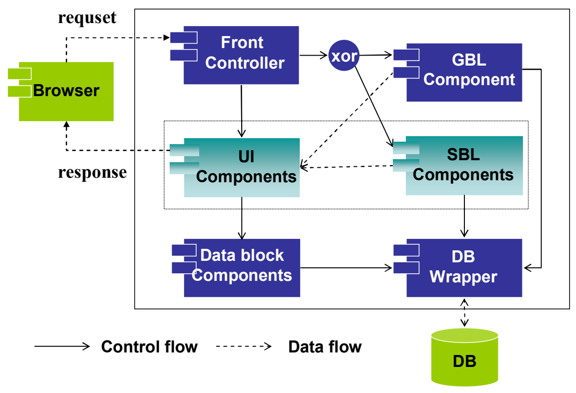

Figure 2. Architecture of Panther Framework

FC orchestrates the application. It receives requests from the browser, interacts with GBLC or SBLC according to the operation type, and then invokes UIC to display an appropriate view to the user. Another important function of FC is to marshalling the requests data into MDO which used by GBLC and SBLC.

GBLC processes the generic operations of the business objects, e.g. CRUD operations. The framework defines 8 kinds of generic operations. Duo to the page limitation, we do not introduce them here. SBLC processes the specific operations of the business object.

DBC is responsible for implementing the UI-R Mapping of the MDO model. It is a large granularity reusable component for assembling the patterned UIC. The DBCs are implemented as custom tags [8]. An extendable DBC library is default provided by Panther, and we can customize more DBCs to improve the power of the framework.

DBW is responsible for implementing the O-R Mapping based on MDO. GBLC, SBLC and DBC use DBW to operate the database.

C. Development with Panther

To demonstrate how to modularize the structure crosscutting concerns and to reduce the clone codes. We introduce the simple guestbook web application that comes from [1]. And due to the page limitation, we concentrate on the implementation of posting messages.

1) Presentation: The new version of insertForm.jsp in Panther framework is shown in Figure 3.

```
<mytag:page>
<mytag:crudForm
  formName= "editForm"
  table=" guestbook "
  keys="id "
  fields=" name,email,message "
  titles=" Your Name|E-mail|Message "
  fieldsInput=" TEXT,TEXT,TEXTAREA"
  operations="INSERT"
  returnPage ="/nat/insert.do" />
</mytag:page>
```

Figure 3. InsertForm.jsp

The JSP file consists of two Tags, one is crudForm which can display the input user interface with freeform style, and the other is page tag which can display the alert information from the functionality tier.

We can see the structure concern (bold and italic lines in Figure 3) is described as meta-data in the crudForm tag. To unweave the structure crosscutting concerns, the HTML generated by the insertForm.jsp also includes the meta-data (bold and italic lines in Figure 4) which will be received by the front controller and marshaled into MDO.

```
<script>
function crudInsert() {
document.editForm.opFlag.value="insert";
document.editForm.submit();
}
</script>
<table><form name="editForm" action="/nat/insert.do">
<input type="hidden" name=" opFlag" >
<input type="hidden" name="fields" value="name,email,message">
<input type="hidden" name="table" value="guestbook">
<input type="hidden" name=" keys" value="id">
<tr><td>Your Name</td><td><input type=text name="name"></td>
</tr>
<tr><td>E-mail</td><td><input type=text name="email"></td>
</tr>
<tr><td>Message</td><td><textarea name="message"></textarea></td>
</tr>
</form></table>
<a href="javascript:crudInsert()">Post</a>
```

Figure 4. HTML generated by insertForm.jsp

```
<web-apps>
...
<app id="/nat/signIn.do"  name="Login verify"
  view=""
  model="nat.panther.sam.handler.SignInHandler">
  <pageFlow forward="Fail" page="/nat/relogin.do"/>
  <pageFlow forward="Success" page="/nat/mainPage.do"/>
</app>
...
<app id="/nat/insert.do" name="Insert Message "
   view="/nat/panther/message/insertForm.jsp" >
   model=""
</app>
...
</web-apps>
```

Figure 5. nat-config.xml

2) Control: As illustrated in Figure 5, the control concern is specified by the nat-config.xml file. This file customizes FC by specifying three mapping relations:

Mapping a physical URI to a SBLC. Every physical URI is specified within the app tag. If there is not SBLC, the model attribute will be set null character.

Mapping a physical URI to a UIC. The mapping information is specified by the view attribute. For example, the /nat/insert.do URI is mapped to /nat/panther/message/insertForm.jsp.

Mapping a logical URI to a physical URI. This mapping can implement the page flow control like the forward attribute in stuts-config.xml[6], we do not discuss this mapping details here.

3) Functionality: In this example, posting message is a generic operation (create operation in CRUD), so we need not write one line code for it. In order to explain the separation of structure concerns from the functionality tier, the value of the operations attribute of the crudForm tag in the insertForm.jsp will be set to "extInsert" which will be considered as a specific operation by the framework, so we can rewrite the default insert operation in the insertAction class.

```
package nat.panther.message;
public class InsertAction extends ActionHandlerSupport {
public processRequest(Action act) throws Exception {
    DBWrapper mydb = new DBWrapper ();
    String opFlag = act.getOpFalg();
    try {
      if(opFlag.equals("extInsert")){
         mydb.add(act.getMDO());
         act.messageBox("Message was successfully added!");
      }
    }catch (Exception e) {
        e.printStackTrace();
        act.messageBox(e.getMessage());
  }
mydb.close();
}
```

Figure 6. InsertAction.java

As illustrated in Figure 6, InsertAction inherits the interface ActionHandlerSupport provided by the framework, and overrides the method processRequest. To separate the structure concern from the InsertAction, the MainServlet passes an Action object to InsertAction, the Action object can marshal the HTTP requests with structure meta-data (see Figure 4) into MDO. Then InsertAction can use DBWrapper to save the message. The Action also can transfer the alert information to JSP files by invoking the messageBox method. The classes diagram is shown in Figure 7.

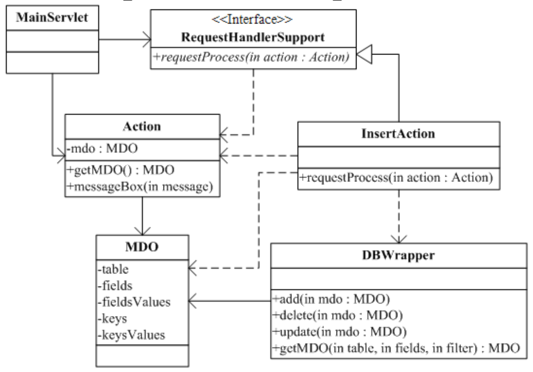

Figure 7. Class Diagram

IV. Related Work

Many approaches have been proposed to improve the adaptiveness and maintainability of the software systems, such as Adaptive Object Model (AOM), Reflection, Aspect Oriented Programming (AOP), UML Virtual Machine (UVM), etc.

AOM also has been called “User Defined Product architecture” and “Dynamic Object Models”. AOM is a system that represents classes, attributes, and relationships as metadata. The system is a model based on instances rather than classes [10]. Users change the metadata to reflect changes in the domain. These changes modify the system’s behavior. In other word, it stores its Object-Model in a database and interprets it. Consequently, the object model is active, when you change it, the system changes immediately. However, AOM requires more effort to build and to learn. And because of its design complexity, it is hard to maintain.

Reflection is a software system’s capability to reason about and act upon itself, adjusting to changing conditions [11]. Reflection can provide objects with the ability to dynamically change their behavior by using design information. However, the reflective systems have two main drawbacks: They offer a too limited set of primitives to develop highly adaptable systems, and they use a fixed programming language.

AOP has been proposed as a mechanism that enables the modular implementation of crosscutting concerns [12]. It provides some mechanisms (join points, pointcut and aspect weaving) that allow of modifying the behavior and the structure of an application, also of a non-stopping application by dynamic weaving. However it has three problems: the first is that the pointcut language is too primitive and not expressive enough, the second is that pointcuts are very tightly coupled to an application’s structure and, the last is that developers are forced to deal with pointcuts at too low level.

UVM is a totally different approach to software development, based on the Model Driven Architecture (MDA). UVM is a runtime environment which will read the UML specification and interpret it on the fly. While the application is running, the UML specification can be changed. New classes, attributes and associations can be added, Algorithmic detail can be added as hand-programmed policy classes that fit into a well-defined extension architecture [13]. The problem is that a UML tool will have to be used to design the new classes, which not every layman will understand. On the other hand, it is imaginable that an application can be created that interfaces with the UML Virtual Machine and exposes a user-friendly GUI to the user to define new classes, associations and behaviors.

V. Discussions

A. Benefits

Two advantages can be achieved by using Panther. One is the high adaptability. The structure concern that was encoded in the program is now modularized in the meta-data (attributes of the custom tag) so that changes to the program lead to changes in the content of the meta-data. For example, changing the existing structure of the guestbook to add a date field, we only need to modify the insertForm.jsp. So the framework can significantly reduce the application maintenance cost.

The other is high reusability. The large granularity, reusable and configurable Custom Tag library makes the web application easy to development. Compare to WebJinn/DDD Framework [1], the lines of code of insertForm.jsp and InsertAction.java in the Panther framework are fewer, and the structure.xml is not needed.

B. *Limitations*

Panther is a domain-specific framework, because the components in the Custom Tag library are only fit for assembly the patterned user interfaces in the web-based EIS. Panther supports to build web application without DBCs, but the high adaptability and reusability can not be achieved.

VI. Conclusion

Crosscutting concerns and the clone codes make the web-based EIS difficult to evolve and maintain. The contribution of this paper is in presenting a new web application development model, namely MDO, to resolve these problems. And a new framework named Panther based on MDO model is developed. The framework can significantly increase development productivity and improve the adaptability of the web application.

An interesting direction for future work is to research the model-driven development (MDD) [9] based on the MDO model. The code structure is very regular, so it very fit for code generation in MDD.

Acknowledgment

Research works in this paper are partial supported by the National Nature Science Foundation of China (Grant No.60773064) and the National Nature Science Foundation of China (Grant No.60673025).

References

[1] S. Kojarski, D. H. Lorenz. Domain driven web development with WebJinn. In Companion of the 18th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications, 2003, pp. 53-65.

[2] Goldberg, D. Robson. Smalltalk-80: The Language and its Implementation. Addison-Wesley, 1983.

[3] T. Kamiya, S. Kusumoto, and K. Inoue. CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code. IEEE Trans. Software Eng., vol. 28, no.7, July 2002, pp. 654-670.

[4] P. J. Molina1, S. Melia1, and O. Pastor. User Interface Conceptual Patterns. Interactive Systems. Design, Specification, and Verification, 9th International Workshop, 2002, pp.159-172.

[5] M. Fowler. Patterns of Enterprise Application Architecture, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2002.

[6] Struts. http://struts.apache.org/.

[7] Spring. http://www.springframework.org/.

[8] B. A. Burd. JSP: JavaServer Pages. Wiley, 2001.

[9] P. Fraternali, P. Paolini. Model-driven development web applications: the AutoWeb system. ACM Transactions on Information Systems, 18(4), 2000, pp.323-382.

[10] J. W. Yoder, R. Johnson, The adaptive object-model architectural style, 3rd IEEE/IFIP Conference on Software Architecture (WICSA3), Kluwer, August 2002, pp. 3–27.

[11] S. Rank, Architectural Reflection for Software Evolution, ECOOP'05 Workshop on Reflection, AOP, and Meta-Data for Software Evolution, 2005, pp51-58.

[12] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. and Irwin, J., Aspect-oriented programming, European Conference on Object-Oriented Programming, 1997, pp.220-242.

[13] D. Riehle, S. Fraleigh, D. Bucka-Lassen, N. Omorogbe. The Architecture of a UML Virtual Machine, Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications, October 2001, p.327-341.